Sample solution

Authors: Mateusz Wykurz, Robert Źrałek

Definition 1. *Travel graph is a graph representing travels which are consistent with given tickets.*

1. *Vertices include information about which city we're in, which ticket did we use to arrive to this city, how many cities from this ticket did we respectively visit and how many cities are still left to visit.*

2. *Edges have costs. There are two types of edges. Those with a zero cost, which lead from one city to another but on the same ticket, assuming we already bought a ticket. Second type are those edges with positive cost, which lead from first to second city for some ticket having cost of this ticket.*

3. *Besides that this graph also contains one additional vertex start which doesn't represent any particular city. No ticket was used to get to it and we still have all cities from a trip to visit. From this vertex lead edges with costs of all available tickets just as start would represent first city on those tickets.*

Theorem 1. *In a travel graph all paths correspond to some trip. Correspondence means that in this trip in all visited cities state, as described in vertices, corresponds to state in vertices on this path and cost of this trip is sum of costs of all edges on this path. Also all possible travels consistent with available tickets have a corresponding path.*

Proof. In a travel graph edges from any vertex lead only to vertices that represent continuation of a travel on a same ticket, which doesn't cost us anything, or buying a new ticket. In both cases edges are present only if possible. Trip can be started from any city and vertex start corresponds to any city, because all ticket edges lead from it.

Corollary 1. *Cheapest possible trip in which we respectively visited all given towns is the cheapest path leading from start to some vertex with number of cities to visit equal zero.*

Algorithm 1. *Scheme of an algorithm used:*

1. *Create travel graph based on available tickets and list of cities to visit.*

2. *Find the cheapest path leading from start to some vertex with number of cities to visit equal zero.*

Correctness: Is straightforward from Theorem 1.

Complexity: Lets denote number of vertices in travel graph by n and number of edges by m. Then creating this graph can be easily done in $O(mlog(n))$ time. Starting from start and building rest of the graph recurrently whenever we want to add edge leading to some vertex we have to check, if it hasn't been already added. All edges are created once and checking if a vertex exists can be done using $Red - Blacktrees$.

Finding the shortest path from start to some vertex with number of cities to visit equal zero can be done using $Dijkstra's algorithm$. Simple implementation works in $O(mlog(n))$ time and we will find shortest paths from start to all vertices. What's left to do is to find shortest of them, which can be done in $O(n)$ time.

Number of vertices in travel graph can easily be estimated by the information vertices contain. Lets denote:

- a = number of cities
- b = number of tickets
- c = number of cities on tickets
- d = number of cities to visit

$n = O(a * b * c * d + 1)$

But a can be deduced from b and c, so in general for some function f and any a, b, c contained in some vertex equation can be written: $f(b, c) = a$. It means that a better valuation can be done:

$n = O(b * c * d + 1)$

From all vertices but start lead at most 1 + number of tickets = 1 + b edges. This gives us:

$m = O(2 * (n - 1) + 20)$

Testing 1. *One of our main goals was to create good tests that would be consistent with the contents of the problem. One of constraints was that only one possible solution for all tests would be possible.*

Using Dijkstra's algorithm on travel graph gives us costs of paths to all vertices. If two such paths have the same length can be easily checked.

But there may be that two different paths, both with minimal cost, lead to same vertex.

Example:
3
10 3 1 2 4
10 3 1 3 4
10 2 4 5
1
3 1 4 5
0

Two different paths lead through tickets 1, 3 and 2, 3 to the same vertex. Both have same minimal cost equal 20.

Algorithm 2. *One that was used to verify if test case has unique solution.*

1. *Create travel graph based on tickets and list of cities to visit.*
2. *Find shortest paths to all vertices with number of cities left to visit equal zero. If there are at least two paths with same minimal cost, or there is none, than test is bad and algorithm ends. Else pick the shortest path and memorize it's cost.*
3. *For each edge on the shortest path:*
   (a) *Remove this edge.*

(b) Find cost of the shortest path from start to some vertex with no cities left to visit.

(c) Compare its cost with memorized cost of shortest path. If they're equal than test is bad. Algorithm ends.

4. Test jest good. Algorithm ends.

Correctness: For two distinct paths in a graph there must exist at least one edge that one of this paths contains and other does not.

After removing any edge from a graph in the new graph there exist only paths that existed in the first one without paths containing the deleted edge.

Conclusion: If there exist two paths both with minimal cost Algorithm 2 will find them.